CSCD01 Deliverable 3

Team 34: 3+3

Bugs & Features:

Issue #691: Add "no value selected" error state on landing home page
Description: When landing on the homepage, there are two payment methods that Mozilla provides to its donors**.** When users simply click on the Visa payment button, the homepage refreshes itself without displaying any visible error prompts, also chooses another currency and repeat the actions above will route back to default currency that is (USD).

Investigation of #691:
The problem occurs in the javascript file
https://github.com/mozilla/donate-wagtail/blob/master/source/js/main.js
when the user clicks on Visa without selecting an amount, it triggers "**document.addEventListener("DOMContentLoaded", function()"** causes the page to refresh, and reload the DOM elements. They didn't create any error message handler for the case that no donate amount of either one-time panel or monthly panel is selected. We could set a default value at the homepage or add error message prompts to avoid reloading the page which causes unnecessary inconveniences.
Create a state variable in the constructor of
https://github.com/mozilla/donate-wagtail/blob/master/source/js/components/donation-amount-toggle.js
we can create a new function to handle the **notAmountSelectedError()** in file
https://github.com/mozilla/donate-wagtail/blob/master/source/js/payments-card.js
If neither panel (one-time, monthly) is selected. This error handler should block the **DOMContentLoaded** event by checking the state variable in the component's constructor.

Acceptance tests:

1. Follow the instruction to set up the docker environment.
2. Visit the page http://0.0.0.0:8000/.
3. Click the button Visa without selecting any donation amount
4. An error should display with the message "please select an amount or type the amount in other amount field " instead of reloading the page.
5. The donate currency should remain the same after the error message has shown

Issue #761: Proper error message when clicking the PayPal button

Description: When the user clicks the PayPal button without selecting an amount to donate for, the page shows an error message saying “There was an error processing your payment. Please try again.”, which gives no context to the user what the actual problem is, and if the user listens and keeps trying again, he will be stuck there seeing this error message repeatedly and not knowing that he should pick an amount to donate for.

Investigation of Issue #761:
The problem happens to be in the file **/source/js/components/paypal.js** where they set up their paypal payment api. The problem occurs because they choose to output “There was an error processing your payment. Please try again.”, which is a general error message that they have set up under an error that was not because of an error when processing the payment, but actually because the user has not selected an amount to donate for.

Acceptance tests:
1: Follow instructions on https://github.com/mozilla/donate-wagtail to set up local dev environment
2: Open up the browser and go to http://localhost:8000/
3: Clicks PayPal button
4: An error at the top should occur with the message “You have not selected an amount to donate for yet, please try again after you select a desired donation amount.” rather than “There was an error processing your payment. Please try again.”

Issue #661: **"Other amount" text field allows amounts in scientific notation**
Description: When the user selects "Other amount" on the front page's donation form, the form accepts scientific notation as a valid input and converts it into an integer. For example, entering the value "1e3" will be parsed as a $1000 donation. Interestingly, the input field rejects any characters except for the "e" character, which allows the scientific notation bug in the first place.

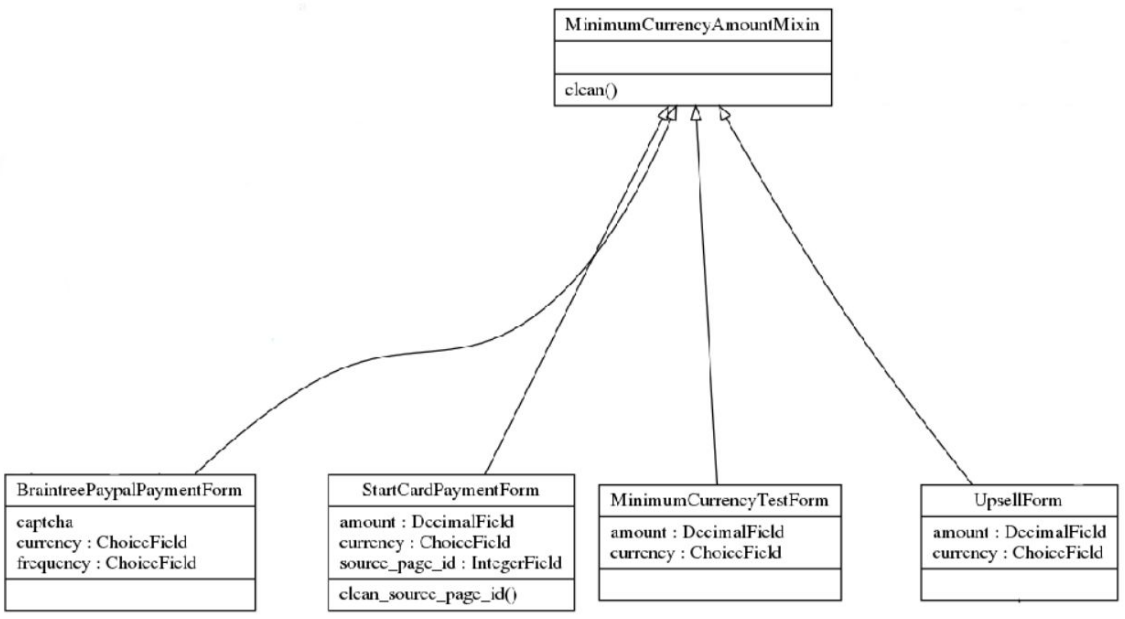
Amounts in scientific notation do not affect the hard-coded donation limit of $10,000,000., so if "1e12" is inputted, a popup notifies the user that the value must be less than or equal to $10,000,000.
Investigation of Issue #661:
There are two places in the code that need to be changed in order to prevent the input of this “e” character, one place in the front-end and one place in the back-end.

On the frontend, there is a form template at https://github.com/mozilla/donate-wagtail/blob/b1806db7330011630f732e8d29faaaeb430c2918/donate/templates/fragments/donate_form.html#L40 that needs to be

modified in order to reject the “e” character, possibly using a Javascript script that runs on form input. This is the template for the One-Time custom payment input form, so the Monthly custom payment input form needs to be modified as well (at https://github.com/mozilla/donate-wagtail/blob/b1806db7330011630f732e8d29faaaeb430c2918/donate/templates/fragments/donate_form.html#L100).

On the back-end, there is a MinimumCurrencyAmountMixin at https://github.com/mozilla/donate-wagtail/blob/b1806db7330011630f732e8d29faaaeb430c2918/donate/payments/forms.py#L22 that needs to be modified in order to reject any input in the form that contains the “e” character. This needs to be changed as well in case a malicious actor attempts to inject this character into the form through a direct cURL request instead of through the actual frontend form.

UML:



We will be modifying the code in the MinimumCurrencyAmountMixin, and plan to test each of its dependencies (BraintreePaypalPaymentForm, StartCardPaymentForm, and UpsellForm). Note also there is also a MinimumCurrencyTestForm that should still run correctly after we modify the code, which will be our regression test.

Acceptance tests:

Acceptance test for one-time payments:

1. Visit the URL http://0.0.0.0:8000/.
2. Click the “One-Time” radio button to attempt a one-time donation.
3. Type “1e3” in the custom amount input box (the input box prefixed with a “$” sign).
4. Choose the Visa payment and click the Visa button.
5. An error should occur, and the user should not be shown the payment detail form and instead be redirected to the front page.

Acceptance test for monthly payments:

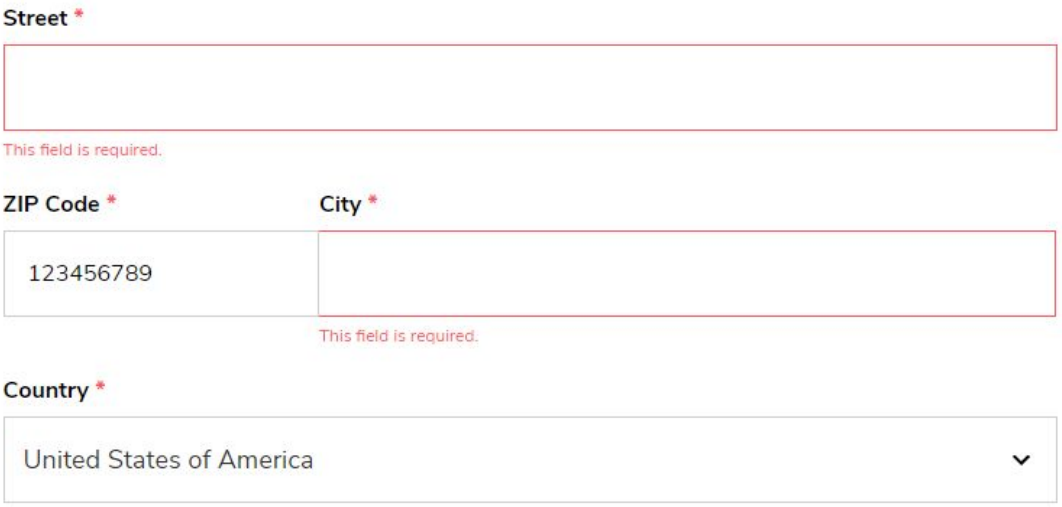
1. Visit the URL http://0.0.0.0:8000/.
2. Click the “Monthly” radio button to attempt a one-time donation.
3. Type “1e3” in the custom amount input box (the input box prefixed with a “$” sign).
4. Choose the Visa payment and click the Visa button.
5. An error should occur, and the user should not be shown the payment detail form and instead be redirected to the front page.

Issue #367: Zip code has no error for invalid input

Description: When given a Zip/Postal code for payment method the field raises no errors with an input that does not match the Zip/Postal code pattern.
This issue was made originally from Issue #343 which is about insufficient error reporting on the frontend. The three issues are the following:

1. Create clear error state of 'Other Amount' <input> #363
2. Email input on cc payment form presents no error for invalid input #366
3. Zip code has no error for invalid input #367

Below is a screenshot of the lack of error logic/handling.



Investigation of Issue #367:
In /donate/payments/forms.py there is no form validator for the CharField post_code. However, even though the zip code is exclusive to just the United States, if you change the country selected in the dropdown, it is still labelled as zip code even if you select Canada for instance.
The very easy solution is just to limit the number of characters in the field to about 12 which should encompass all the postal codes in the world and anything longer would be invalid. This is a one-liner change in /donate/payments/forms.py.

The more tricky solution would be to validate the “Zip code” field depending on the country selected in the dropdown field. We would have to create a clean_post_code(self) method within the BrainTreeCardPaymentForm class in forms.py. It will have to validate whether the zip code matches whatever corresponding postal code for the selected country.

Acceptance tests:

1. Follow instructions on setting up dev environment
2. Open up browser and go to http://localhost:8000/
3. Select amount to donate and click on VISA card payment button
4. Entering ‘1234567890’ into the zip code field and clicking the donate button should give an error for the zip code field.

Issue #846: **Enable ApplePay, GooglePay, and SamsungPay in Braintree**
Description:
Investigation of Issue #846:
ApplePay, GooglePay and SamsungPay have changed how companies manages modern sales pipelines. They allow for automatic filling of payment forms from a browser or phones cache. A large number of sales are lost when users are prompted to input their payment details, ApplePay, GooglePay and SamsungPay fix this by allowing sites to autofill this information. Implementing this feature would see a massive increase in Mozilla donations through the site. Braintree has API for these payment methods it is just a matter of connecting the API with the form and testing it for security purposes

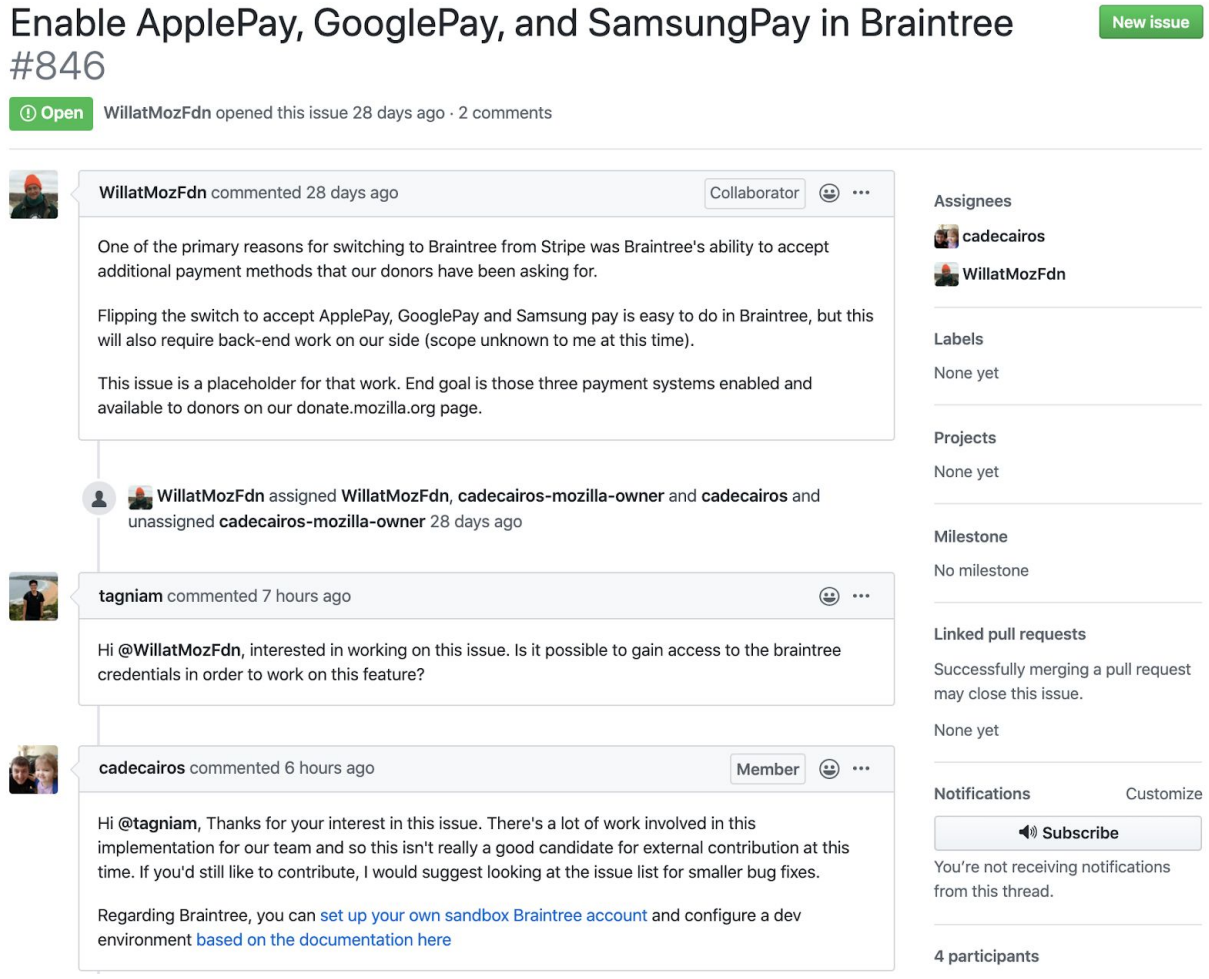
Code Base to Modify

All the changes are to the payment system that is located in the payments folder. The following Files will have to be edited
/donate/payments

- tasks.py
  This is the main file for the payments system including all the payment handlers a new handler will have to implement to pull data from ApplePay, GooglePay and SamsungPay. In addition this would need to add to the successful payment hander and the unsuccessful payment handler
- forms.py
  This is the file that handles forms for submitting payment methods, this would have to import the payment system api and pass them along to braintrees api to handle the request. Buttons and frontend elements for handling this process would be created here.

Reasons for selecting four small bugs instead of the issue #846

For deliverable 3, our team browsed through the lists of all the opening issues for acceptable features/fixes and came up with issue #846. We noticed that issue #846 may need access to the Braintree credentials and left a comment asking for help under the issue description. And one of the members from the Mozilla Team suggests that issue #846 is not a good candidate for external contribution since it involved a lot of work for their team.



Since there are only 73 opening issues in the project repository, we could not find another doable issue to work on for deliverable 3. After carefully considering the workload and difficulty of other issues in the repo and the conversation with the TA, we decided to go with the issues we did not implement in deliverable 2.

Architecture of the System

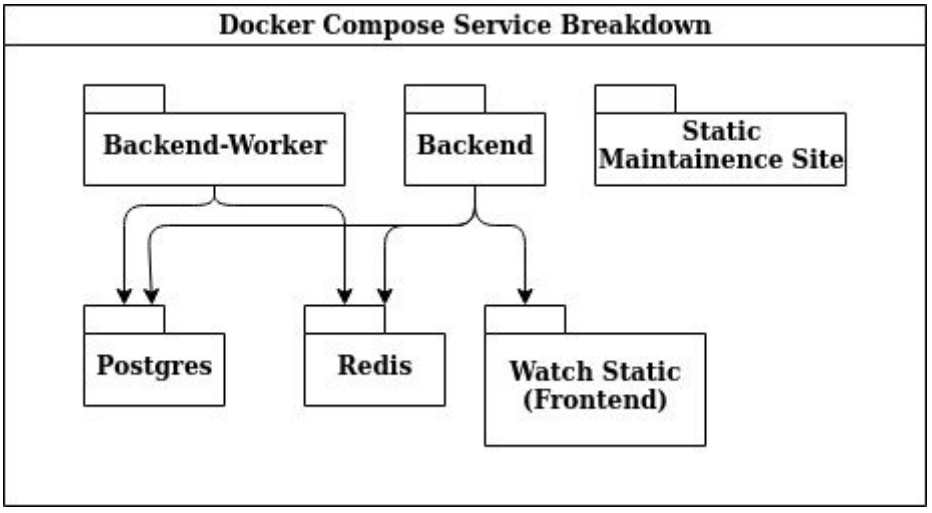


Figure 1: Docker Compose Service Breakdown.

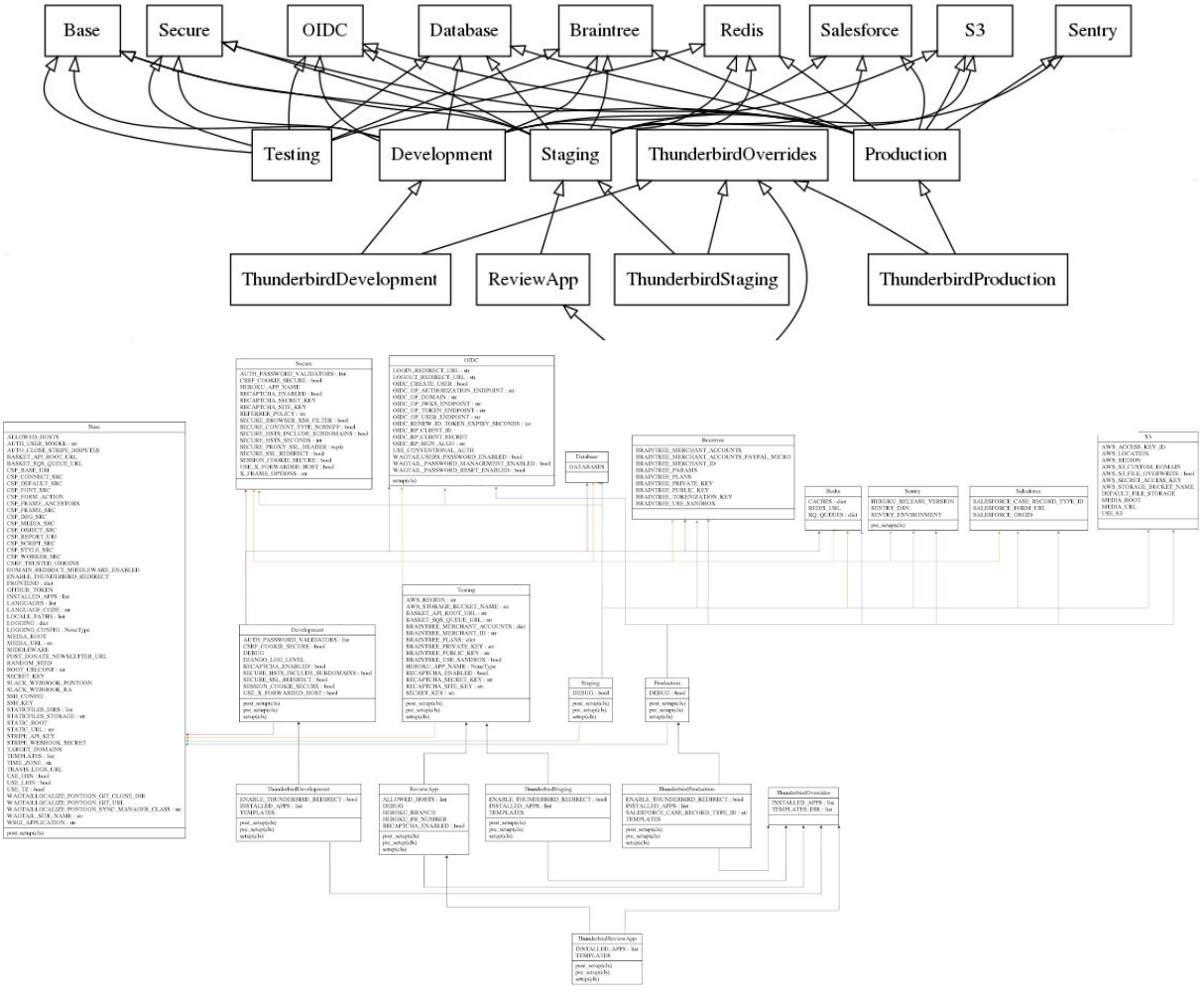


Figure 2: Backend Django configurations and dependencies.

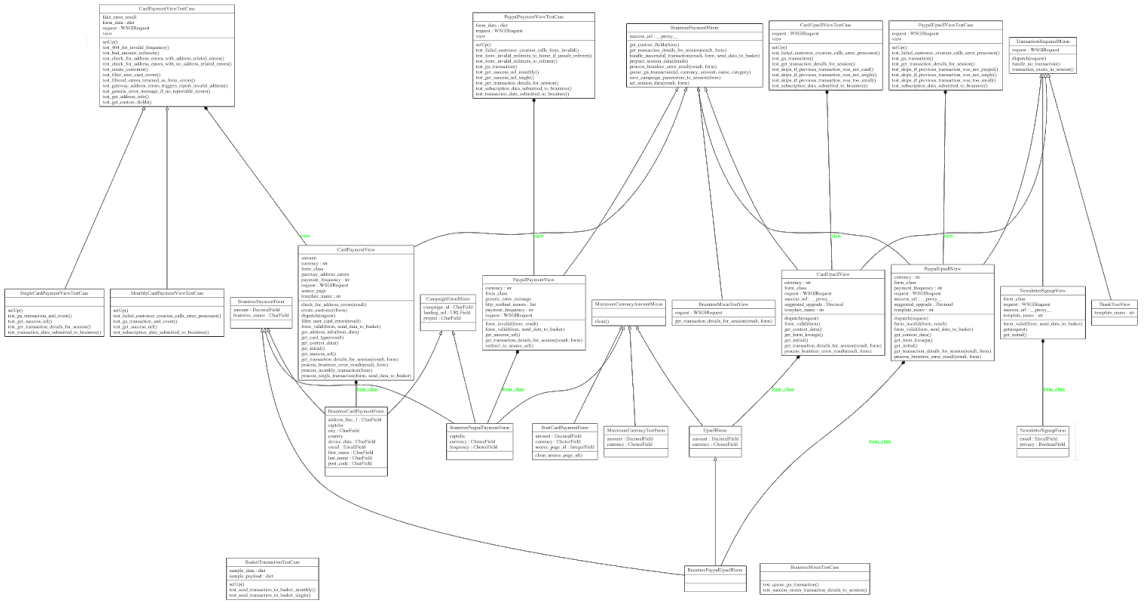


Figure 3: Payment system overview.